A MPBSG Technique Based Parallel Dual-Type Method Used for Solving Distributed Optimal Power Flow Problems

Huay CHANG†, Member and Shieh-Shing LIN††, Nonmember

SUMMARY In this paper, we propose a method to solve the distributed optimal power flow problem and discuss the associated implementation. We have combined this method with a projected Jacobi (PJ) method and a modified parallel block scaled gradient (MPBSG) method possessing decomposition effects. With the decomposition, our method can be parallel processed and is computationally efficient. We have tested our method for distributed OPF problems on numerous power systems. As seen from the simulation results, our method achieved a dramatic speed-up ratio compared with the commercial IMSL subroutines.

key words: distributed optimal power flow, parallel dual-type method, projected Jacobi, quadratic programming

1. Introduction

In this paper, we use the Parallel Dual-type Method based on MPBSG Technique to solve the Distributed Optimal Power Flow Problems. In this world, most of the large practical systems are formed by interconnected subsystems. For examples, the power system is formed by area-like subsystems. These subsystems are interconnected with each other through the tie-lines. The ground transportation system uses the highway system to interconnect with the local transportation system. The telephone network contains local networks and long distance network. The telephone network is used to interconnect with different local networks. Those network systems become more difficult in operation management and status control. Conventionally, a central control center is employed to manage the operations of the whole system. Nowadays, since computer communication technologies have become more matured. The decentralized management and control grows up toward the current deregulated large power system [1] which is formed by area-like subsystems through tie lines.

The nonlinear large network optimization problem, such as optimal power flow (OPF) problem, encounters the computational difficulty. The reasons for the computation difficulty are: their large dimensions and nonlinearity. Most of the solution techniques have been developed. And those solutions are based on nonlinear programming techniques. Several papers [2]–[12] have mentioned about these solutions. In [13], a distributed implementation was described. The network of the workstations is used to solve the decentralized OPF problems. As to the conventional parallel approach, the usage of Lagrange Relaxed method is to solve the distributed OPF within inequality constraints. There is a method described in [14] and [15]. The method submitted in [14] and [15] is to use Lagrange multiplier to relax the constraints. This method also uses the KKT condition to solve the problem. But if there are more functional inequality constraints in the sub systems, the method might become more complicate and more difficult while solving the distributed OPF. Presenting methods to deal with a large-scale of optimal power flow problem for large distributed systems in a distributed computer network is the purpose of this paper.

The paper is organized in the following structures: Sect. 2 states the problem of the distributed OPF problem. Section 3 presents the method combing projected Jacobi and parallel dual-type method for solving distributed OPF problems. The simulation results those are used to demonstrate the computational efficiency are given in Sect. 4. At last, Sect. 5 gives a brief conclusion, and the MPBSG technique is given in Appendix.

2. Statement of the Distributed Optimal Power Flow Problem

We first introduce the notations for the $i$th area or subsystem $(i = 1, 2, \ldots, n)$ which are following from [16].

- $e_i, f_i$: states variables represent the real and imaginary part of the complex voltage.
- $u_i$: control variables including real and reactive power generation, $P_{Gi}$, and $Q_{Gi}$, transformer tap ratio, switching capacitor banks, etc.
- $x^i: = (u_i, e_i, f_i)$ denotes the vector of all variables in area $i$.
- $x^i_h$: denotes the boundary vector of the other subsystems connecting with subsystem $i$.
- $F_i(x)$: objective function which can be total generation cost, pollution cost, system losses, etc.
- $g_i(x)$: real and reactive power mismatch.
- $h_i(e_i, f_i)$: functional inequality constraints, such as security constraints on line flows for specified lines.
- $V_i$: vector of voltage magnitude, $V_i = \sqrt{e_i^2 + f_i^2}$.
- $\bar{V}_i, \underline{V}_i$: upper and lower bound of voltage magnitude.
- $\bar{u}_i, \underline{u}_i$: upper and lower bound of control variables $u_i$, such as $P_{Gi}, Q_{Gi}, \bar{Q}_{Gi}$, etc.

Copyright © 2006 The Institute of Electronics, Information and Communication Engineers
\( \bar{h}_i, \underline{h}_i \): upper and lower bound of functional inequality constraints.

\( k, t, w \): iteration index.

\( \text{diag} \[\square\] \): a diagonal matrix formed by the diagonal terms of the matrix \( \square \).

\( \alpha, \beta, \gamma \): step-size.

\( d(\cdot) \): the increment of the vector \( (\cdot) \).

\( \lambda_i \): the Lagrange multiplier vector.

\( \Pi \): quadratic programming.

\( \phi_i(\lambda) \): the dual function of the \( i \)th subproblem.

\( \Gamma_i \): the set formed by the inequality constraints of the \( i \)th subproblem.

\( \eta_i, \delta, \varepsilon \): positive real numbers.

The distributed OPF problem can be stated as follows:

\[
\min_{x} \sum_{i=1}^{n} F_i(x_i, x_i^0) \\
g(x) = 0 \\
h_i \leq h \leq \bar{h}_i \\
V_i \leq \bar{V}_i \\
\underline{u}_i \leq u_i \leq \bar{u}_i \quad i = 1, 2, \ldots, n. \tag{1}
\]

Let the objective function, the real state vector, the real and reactive power mismatch, the functional inequality constraints, the voltage constraints, and the control variable constraints of the whole system be denoted by \( F, x, g, h, v, u \), respectively. Then

\[
F = \sum_{i=1}^{n} F_i, \quad x^T = [x_1^T, x_2^T, \ldots, x_n^T],
\]

\[
g^T = [g_1^T, g_2^T, \ldots, g_n^T], \quad h^T = [h_1^T, h_2^T, \ldots, h_n^T],
\]

\[
V^T = [V_1^T, V_2^T, \ldots, V_n^T], \quad u^T = [u_1^T, u_2^T, \ldots, u_n^T].
\]

Based on Eq. (1), the distributed OPF problem of the whole system can be expressed as:

\[
\min_{x} F(x) \\
g(x) = 0 \\
h \leq h \leq \bar{h} \\
V \leq \bar{V} \\
u \leq u \leq \bar{u}. \tag{2}
\]

3. Solution Method

3.1 The PJ (Projected Jacobi) Method

The PJ method uses the following iterations to solve the distributed OPF problem given in Eq. (2),

\[
x(k + 1) = x(k) + \alpha(k)d^*x(k), \tag{3}
\]

where \( \alpha(k) \) is a step-size determined according to centralized Armijo’s rule [17], and \( d^*x(k) \) is the solution of the following QP subproblem:

\[
\min_{dx} \nabla F^T(x(k))dx + \frac{1}{2} dx^T H dx
\]

where the matrix \( H =
\[
\begin{bmatrix}
\nabla^2 F_1(x_1, x_1^0) & 0 & \cdots & 0 \\
0 & \nabla^2 F_2(x_2, x_2^0) & \cdots & \vdots \\
\vdots & \cdots & \cdots & 0 \\
0 & \cdots & 0 & \nabla^2 F_n(x_n, x_n^0)
\end{bmatrix}
\]

\[
+ \frac{1}{2} \delta I
\]

and the \( H_0 \) is the block diagonal submatrices of \( H \) in which \( \nabla^2 F_i(x) \) is the Hessian of \( F_i(x) \) and \( \delta \) is positive real number as where \( I \) is an identity matrix, and \( \delta \) is a small positive real number but enough to make \( H \) positive definite.

From Eq. (1) and Eq. (2), and since \( H \) is a diagonal matrix, we can rewrite Eq. (4) as

\[
\min_{dx} \sum_{i=1}^{n} \nabla x_i F_i^T(x(k))dx_i + \frac{1}{2} dx_i^T H dx_i
\]

\[
g(x(k)) + \nabla g_i^T(x(k))dx_i = 0 \\
h_i \leq h_i \leq \bar{h}_i \\
V_i \leq \bar{V}_i \\
u_i \leq u_i \leq \bar{u}_i \quad i = 1, 2, \ldots, n. \tag{5}
\]

From the above descriptions, the iterative update Eq. (3) of PJ method can be decomposed into

\[
x_i(k + 1) = x_i(k) + \alpha(k)dx_i^*(k) \quad i = 1, 2, \ldots, n. \tag{6}
\]

3.2 The Parallel Dual-Type (PDT) Method for Quadratic Programming Problem

The difficulties encountered in solving a constrained optimization problem using primal method [17] are (i) the need to project the seeking direction onto the surface formed by the equality and inequality constraints [17], and (ii) the need to determine an initial feasible point. The projection usually requires huge computations and, most of the times, limits the convergence speed of the algorithm. Thus, to circumvent such kind of difficulty occurred in primal methods, the dual-type methods that solve the dual problem become attractive.

In the following, we will first formulate the corresponding dual problem of Eq. (5) and then present the proposed parallel dual-type method to solve it.

The dual problem of the QP subproblem Eq. (5) is

\[
\max_{\lambda} \phi(\lambda), \tag{7}
\]

where the dual function
\[
\phi(\lambda) = \min_{dx \in \Gamma} \sum_{i=1}^{n} \nabla_{x_i} F^T_i(x(k))dx_i + \frac{1}{2} dx^T H dx + \lambda^T [g_t(x(k)) + \nabla_{x_t} g^T_t(x(k))dx_t].
\]

(8)

\(\Gamma\) denotes the set of inequality constraints in Eq. (5), such that

\[
\Gamma \equiv \{dx \mid h_i(k) + \nabla e_i h^T_i(k)de_i + \nabla f_i h^T_i(k)df_i \leq \bar{h}_i, \quad V_i \leq V_t(k) + \nabla e_i V^T_i(k)de_i + \nabla f_i V^T_i(k)df_i \leq \bar{V}_i, \quad u_i \leq u_t(k) + du_t \leq \bar{u}_i, i = 1, 2, \ldots, n\}.
\]

The parallel dual-type method uses the following iterations to solve Eq. (7),

\[
\lambda_i(t+1) = \lambda_i(t) + \beta(t) d\lambda_i(t), \quad i = 1, 2, \ldots, n
\]

(9)

where \(t\) is the iteration index, the positive scalar \(\beta(t)\) is a step-size determined according to centralized Armijo’s rule [17], and the increment of the Lagrange multiplier \(d\lambda^T(t) = [d\lambda_1^T(t), \ldots, d\lambda_n^T(t)]\) is the solution of the following approximate quadratic problem of Eq. (7) at \(\lambda(t):\)

\[
\max_{dx} \frac{1}{2} dx^T \Phi dx + \nabla \phi^T dx
\]

(10)

The matrix \(\Phi\) in Eq. (10) has dimension \(m \times m, m = m_1 + m_2 + \cdots + m_n\), and is given by

\[
\begin{bmatrix}
\Phi_1 & 0 & \cdots & 0 \\
0 & \Phi_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & 0 \\
0 & 0 & \cdots & \Phi_n
\end{bmatrix}
\]

(11)

where the \(m \times m\) block submatrix \(\Phi_i\) can be obtained by

\[
\Phi_i = -\nabla_{x_i} g^T_i(x(k))H_i^{-1} \nabla_{x_i} g_i(x(k)),
\]

(12)

in which \(H_i\) is the \(i\)th diagonal block submatrix of \(H\) corresponding to subsystem \(i\). The derivative of the dual function, \(\nabla \phi\), in Eq. (10) can be expressed as \(\nabla \phi^T(\lambda(t)) = [\nabla_{\lambda_1} \phi^T(\lambda(t)), \ldots, \nabla_{\lambda_n} \phi^T(\lambda(t))]\), and can be computed by

\[
\nabla_{\lambda_i} \phi(\lambda(t)) = g_t(x(k)) + \nabla_{x_t} g^T_t(x(k))d\lambda_t(x(k)),
\]

(13)

where \(d\lambda^T(\lambda(t)) = [d\lambda_1^T(\lambda(t)), \ldots, d\lambda_n^T(\lambda(t))]\) is the solution of the minimization problem on the RHS of Eq. (8) [16].

Decomposition Effect. Since \(H_i\) is positive definite and \(\nabla_{x_t} g_t(x(k))\) is of full rank, the \(\Phi_i\) given in Eq. (11) as well as \(\Phi\) given in Eq. (10) should be negative definite. Therefore, the objective function in Eq. (10) is concave. Consequently, \(d\lambda_t(\lambda(t))\), the solution of the approximate quadratic dual problem Eq. (10) can be obtained by solving the following optimal necessary condition of Eq. (10) [17]

\[
\Phi_i d\lambda_t(\lambda(t)) = -\nabla \phi_t(\lambda(t)),
\]

(14)

which can be decomposed into the following \(n\) independent sets of linear equations

\[
\Phi_i d\lambda_t(\lambda(t)) = -\nabla \phi_t(\lambda(t)), \quad i = 1, 2, \ldots, n.
\]

(15)

These \(n\) sets of Eq. (15) can be executed in parallel in a distributed computer network if each \(\nabla_{x_t} \phi_t(\lambda(t))\) is obtained. In fact, it is the decomposition effect that makes our dual-type method becoming parallel. In addition, this effect also contributes to the computational efficiency of our method.

To compute \(\nabla_{x_t} \phi_t(\lambda(t))\), we need to solve the minimization problem on the RHS of Eq. (8) to obtain \(d\lambda^T(\lambda(t)) = [d\lambda_1^T(\lambda(t)), \ldots, d\lambda_n^T(\lambda(t))]\). This can be achieved by using the following two-phase algorithm [16].

Phase 1: Solve the following unconstrained minimization problem:

\[
\min_{dx} \sum_{i=1}^{n} [\nabla_{x_i} F^T_i(x(k))dx_i + \frac{1}{2} dx^T H dx + \lambda^T [g_t(x(k)) + \nabla_{x_t} g^T_t(x(k))dx_t]]
\]

(16)

by using a modified parallel block scaled gradient method [18] to obtain a solution,

\[d\lambda^T(\lambda(t)) = [d\lambda_1^T(\lambda(t)), \ldots, d\lambda_n^T(\lambda(t))],\]

this method is briefly described in Appendix.

Note that the Eq. (16) is the unconstrained minimization problem on RHS of Eq. (8) but without the constraints \(dx \in \Gamma\).

Phase 2: Project \(d\lambda_t(\lambda(t))\), the solution obtained from phase 1, onto \(\Gamma\). The resulting projection is \(d\lambda_t(\lambda(t))\). It is the solution of the minimization problem on RHS of Eq. (8). In phase 2, the constraint set \(\Gamma = \cup_{i=1}^{n} \Gamma_i\), where

\[
\Gamma_i \equiv \{dx \mid h_i(k) + \nabla e_i h^T_i(k)de_i + \nabla f_i h^T_i(k)df_i \leq \bar{h}_i, \quad V_i \leq V_t(k) + \nabla e_i V^T_i(k)de_i + \nabla f_i V^T_i(k)df_i \leq \bar{V}_i, \quad u_i \leq u_t(k) + du_t \leq \bar{u}_i\}
\]

(17)

and the resulting projection \(d\lambda_t(\lambda(t)), \ldots, d\lambda_n(\lambda(t))\) can be computed by

\[
d\lambda_t(\lambda(t)) = \begin{cases} 
\bar{x}_i - x_t(\lambda(t)) & \text{if } x_t(\lambda(t)) + d\lambda_t > \bar{x}_i \\
\bar{x}_i - x_t(\lambda(t)) & \text{if } x_t(\lambda(t)) + d\lambda_t < \bar{x}_i \\
\bar{x_t} & \text{otherwise}
\end{cases}
\]

(18)

Note that the computations in phase 2 are the comparison check as shown in Eq. (18). As a similar proof given in [16], it can be easily shown that \(d\lambda_t(\lambda(t)) = [d\lambda_1(\lambda(t)), \ldots, d\lambda_n(\lambda(t))]\) obtained from the two-phase algorithm is the solution of the minimization problem on the RHS of Eq. (8).

Remark 1: \(\Phi\) shown in Eq. (10) is negative definite matrix, because each \(\Phi_i\) is negative definite matrix in Eq. (11). Thus \(d\lambda(t)\) obtained from Eq. (14) will be an ascent direction for the dual problem Eq. (7). Consequently, with a suitable choice of the step-size \(\beta\) convergence of the parallel dual-type method is guaranteed [19].

3.3 The Complete Algorithm for Solving Distributed Optimal Power Flow Problems

Our method for solving Distributed OPF in Eq. (1) is using the PJ method Eq. (3) where \(dx'(k)\) is the solution of...
the QP subproblem Eq. (4). The proposed parallel dual-type method uses Eq. (9) to solve Eq. (7), the parallel dual problem of the QP subproblem, instead of solving Eq. (5) directly. The \(d \lambda_i(t)\) in Eq. (9) is obtained from solving Eq. (15) using sparse matrix technique and can be executed in a distributed computer network. The \(d \hat{x}_i\) is needed to set up \(\nabla_\lambda \phi(\lambda(t))\) and can be computed using the two-phase method. Consequently, the parallel dual-type method converges to optimal solution \(\lambda^*\) and the solution \(d \hat{x}\) of the constrained minimization problem on the RHS of Eq. (8) with \(\lambda = \lambda^*\) is \(dx\), the solution of Eq. (4).

Now we are ready to state our method for solving Distributed OPF problems.

For every area \(i, i = 1, 2, \ldots, n\):

Step 0: Set the initial step-sizes value \(\alpha, \beta, \gamma\) and set the value of \(\delta\) in \(H\).

Step 1: Initially guess \(x_i(0)\) and set \(k=0\).

Step 2: Initially guess \(\lambda_i(0)\) and set \(t=0\).

Step 3: Using the MPBSG technique in phase 1 to compute \(d \hat{x}_i(\lambda(t))\).

Step 4: Using the projection technique in phase 2 to calculate \(d \hat{x}_i\) by Eq. (18).

Step 5: Compute \(\Phi_i\) and \(\nabla_\lambda \phi(\lambda(t))\) by Eqs. (12) and (13), respectively.

Step 6: Solve \(d \lambda_i(t)\) by Eq. (15).

Step 7: If \(\|d \lambda_i\|_{\infty} < \varepsilon\), go to Step 8; otherwise, update \(\lambda_i(t + 1) = \lambda_i(t) + \beta(t)d \lambda_i(t)\), set \(t = t + 1\) and return to Step 3.

Step 8: If \(\|d \hat{x}_i\|_{\infty} < \varepsilon\), stop and \(x(k)\) is the solution; otherwise, go to Step 9.

Step 9: Update \(x_i(k + 1) = x_i(k) + \alpha(k)d \hat{x}_i(k)\), set \(k = k + 1\) and return to Step 2.

4. Test Results

We have tested our method on numerous distributed OPFs problems of the IEEE 30-bus system and IEEE 118-bus system. We assumed that the IEEE 30-bus system and the IEEE 118-bus system consist of three and four subsystems, as shown in Fig. 1 and Fig. 2, respectively. Each subsystem is indicated by a closed dashed contour. There are two ob-
jective functions for all the tested distributed OPF problems. The first one is the minimum total generation cost criteria as $\sum_G a_i P_{G_i} + b_i P_{G_i} + c_i$, where $P_{G_i}$ denotes the total real power generation bus in area $i$. The $a_i, b_i$, and $c_i$ are the coefficients of the cost curve of $P_{G_i}$. The second one is the minimum total system losses criteria as $\sum_{Li} P_{Li}$ where $P_{Li}$ denotes the total real power loss of transmission line in area $i$ [20].

We considered four sets of different cases with different numbers of nonlinear equality constraints and simple bounded inequality constraints. Now, we formed 16 OPF problems of the form Eq. (1). Each OPF problem resulted from a combination of two objective function criteria, four cases of different number of nonlinear equality constraints and simple bounded inequality constraints, and two different types of testing system of IEEE 30-bus and IEEE 118-bus.

We have made two types of tests by using our method. The first one was to test the computational efficiency of the sequential version of our method in a sequential Sparc-20 workstation. We set the following parameters: $\epsilon = 10^{-3}$, $\delta = 0.01$, and all computers began running from a flat start with initial voltages being $e_i = 1.0$, and $f_i = 0.0$ for all buses in area $i$. We then applied our method to solve these 16 OPF problems in a Sparc-20 machine. The corresponding CPU times and final objective value are shown in Tables 1–4.

Each table consists of four OPFs corresponding to four

| Table 1 | The comparison of the computational efficiency of the sequential version of our method with IMSL subroutines in solving optimal power flow problems on cases (1a)–(4a) of the IEEE 30-bus system using the total generation cost as the objective function. |
|---|---|---|---|---|---|
| Cases | No. of equality constraints | No. of inequality constraints | Our method | IMSL | Speed up ratio |
| (1a) | 20 | 45 | 0.0784 | 154.6541 | 27.2017 | 154.6424 | 150.12 |
| (2a) | 25 | 50 | 0.0891 | 154.6582 | 27.7969 | 154.6671 | 149.21 |
| (3a) | 30 | 65 | 0.0830 | 154.6872 | 29.1291 | 154.6854 | 151.32 |
| (4a) | 41 | 80 | 0.0884 | 154.6891 | 29.8784 | 154.6871 | 148.28 |

| Table 2 | The comparison of the computational efficiency of the sequential version of our method with IMSL subroutines in solving optimal power flow problems on cases (5a)–(8a) of the IEEE 30-bus system using the total system losses as the objective function. |
|---|---|---|---|---|---|
| Cases | No. of equality constraints | No. of inequality constraints | Our method | IMSL | Speed up ratio |
| (5a) | 20 | 45 | 0.0818 | 5.4234 | 29.8225 | 5.4214 | 151.23 |
| (6a) | 25 | 50 | 0.0847 | 5.4312 | 32.1740 | 5.4420 | 149.23 |
| (7a) | 30 | 65 | 0.0847 | 5.4545 | 30.3014 | 5.4482 | 150.23 |
| (8a) | 41 | 80 | 0.0885 | 5.4703 | 31.8537 | 5.4792 | 148.71 |

| Table 3 | The comparison of the computational efficiency of the sequential version of our method with IMSL subroutines in solving optimal power flow problems on cases (1b)–(4b) of the IEEE 118-bus system using the total generation cost as the objective function. |
|---|---|---|---|---|---|
| Cases | No. of equality constraints | No. of inequality constraints | Our method | IMSL | Speed up ratio |
| (1b) | 120 | 210 | 0.1046 | 2638.6654 | NA | NA | NA |
| (2b) | 150 | 240 | 0.1049 | 2638.6912 | NA | NA | NA |
| (3b) | 180 | 270 | 0.1056 | 2638.6923 | NA | NA | NA |
| (4b) | 210 | 300 | 0.1087 | 2638.7138 | NA | NA | NA |

| Table 4 | The comparison of the computational efficiency of the sequential version of our method with IMSL subroutines in solving optimal power flow problems on cases (5b)–(8b) of the IEEE 118-bus system using the total system losses as the objective function. |
|---|---|---|---|---|---|
| Cases | No. of equality constraints | No. of inequality constraints | Our method | IMSL | Speed up ratio |
| (5b) | 120 | 210 | 0.1012 | 13.3207 | NA | NA | NA |
| (6b) | 150 | 240 | 0.1017 | 13.3217 | NA | NA | NA |
| (7b) | 180 | 270 | 0.1012 | 13.3536 | NA | NA | NA |
| (8b) | 210 | 300 | 0.1049 | 13.3589 | NA | NA | NA |
cases of different numbers of nonlinear equality constraints and simple bounded inequality constraints, under a given criteria. In order to achieve the purpose of comparison, we also have solved the same OPF problems with the same setup by the IMSL constrained nonlinear programming subroutines. We used IMSL subroutines to verify our solution by running the same problem with same initial guess. IMSL subroutines is a nonlinear programming tool implemented by the well-known Han-Powell algorithm [17], and the final objective value and the corresponding CPU time consumption for each OPF are also reported in Tables 1–4.

The test results shown in Tables 1–4 corresponding to the cases (4a), (8a), (4b) and (8b) that with many numbers of nonlinear equality constraints and simple bounded inequality constraints, this the resulting CPU times of our method are least compared with the other cases of same criteria. The final objective values appends similar on each case in Table 1, also in Table 2. The dramatic computational efficiency of our method can be observed from the column of the average CPU time consumption in Tables 1 and 2, which shows the speed-up ratio of the sequential version of our method versus IMSL subroutines is around 150 times in solving OPFs problems on the IEEE 30-bus system. However, the IMSL subroutines can not solve the examples on the IEEE 118-bus system because of the huge memories requirement. In addition, our method is more suitable in handling large-scale system, because we successfully employ decomposition technique and sparse matrix technique in our method which results in the reduction of memory requirements and the increase of the computational speed. As we can see in Tables 3 and 4, due to the memory shortage, the IMSL subroutines can not solve the OPFs problems on the IEEE 118-bus system.

To appreciate more about our method that is shown in Fig. 3 and the OPF problem on case (4b), in Table 3. Each black circle in the figures represents single iteration of the PJ method. The CPU time consumed between circles represents the CPU time consumed by the proposed parallel dual-type method for solving Eq. (4), completely. We also indicate the number of iterations of the parallel dual-type method used in single iteration of PJ method in Fig. 3. While \( k = 2 \), the objective function decreased apparently in Fig. 3. Therefore, we can tell that our method inherit a good computational efficiency in Fig. 3.

The second type of these tests we have made is to test the computational efficiency of our parallel (distributed) algorithm. We have already demonstrated the superiority of the sequential version of our method over the IMSL subroutines. Next stage, we show the computational efficiency of our method with another method based on parallel approach. To demonstrate the computational efficiency of the parallel version of our method in a real dedicated network environment, we chose a PC-network of four Pentium IV PCs as our experimental computer network. Each PC is installed with a network adaptor card (communication hardware) which consists of software support for an application program interface and transmission support required to communicate across the network [21]. Figure 4 shows the experimental PC network of four Pentium IV PCs.

![Experimental PC network](image-url)
We use our method to solve the distributed OPF problem by using the total system losses as the objective function on the IEEE 118-bus system. Each PC in Fig. 4 contains the data of the corresponding subsystem and the data of the boundary buses in neighboring subsystems as indicated in Fig. 2. The result of the average computation time and the final objective value for the 1000 OPFs problems in each case, on the IEEE 118-bus system are shown in Table 5. The parallel approach of conventional Lagrange method submitted in [15] is used to implement the comparison of the computational efficiency of the parallel version by using our method (Parallel Dual-type method). In [15], they presented a decentralized implementation of the DC Optimal Power Flow (OPF) on a network of workstation. They used Lagrange multiplier to relax the constraints and used KKT condition to solve the problem. We abbreviate this method as KKT. We also solve the same problems with the same initial guess and same stop criteria in each case using the KKT method, and the average computation time in each case and the final objective value are also shown in Table 5. In Table 5, we see that the average speed-up ratio of the parallel version by using our method represents better efficiency of our method. As we have indicated in section 3.2, the structure of \( \Phi \) is shown in Eq. (11). That structure allows us to decompose the large-dimension linear equation Eq. (15). We also use the MPBSG technique to solve the linear equation Eq. (16) in phase 1.

\[
dx = -H^{-1} \{ \nabla_{x} F(x(k)) + \nabla_{x} g_{i}(x(k))\lambda_{i} \}.
\]

Furthermore, we use the projection technique in phase 2 to obtain \( \delta x_{i} \). After that, we compute \( \Phi_{l} \) and \( \nabla_{x} \phi(\delta t) \) by Eqs. (12) and (13). Then, we also compute \( \delta \lambda(t) \) by Eq. (15). These steps are shown in Step 5 and Step 6. Consequently, the parallel dual-type method converges to optimal solution \( \lambda^{*} \) and the solution \( \delta \Phi \) of the constrained minimization problem on the RHS of Eq. (8) with \( \lambda = \lambda^{*} \) is \( dx \), the solution of Eq. (4).

Since \( H_{ii} \) is block diagonal sub matrices of \( H \), and

\[
H = \text{diag}[\nabla^{2} F(x)] + \frac{1}{2} \delta I,
\]

in which \( \nabla^{2} F(x) \) is the Hessian of \( F(x) \), and \( \delta \) is a small positive real number but it is enough to make \( H \) positive definite,

\[
H = \begin{bmatrix}
H_{11} & 0 & \cdots & 0 \\
0 & H_{22} & \cdots & \vdots \\
\vdots & \vdots & \ddots & 0 \\
0 & \cdots & 0 & H_{nn}
\end{bmatrix},
\]

and

\[
dx = [dx_{1}, dx_{2}, \ldots, dx_{n}],
\]

if we consider the whole system as an area, then our method is similar to the Modifications of Newton Method with approximately quadratic convergence. [17, Chap.7, Sect. 7.8]. Furthermore, there are two additional reasons of the computational efficiency of our method. As we have indicated in section 3.2, the structure of \( \Phi \) is shown in Eq. (11). That structure allows us to decompose the large-dimension linear equation Eq. (14) into \( n \) independent sets of small-dimension linear equation Eq. (15). We also use the MPBSG technique to solve the linear equation Eq. (16) in phase 1. These two major factors contribute to the computational efficiency of our method.

5. Conclusions

We have presented a parallel dual-type method for solving distributed OPF problems. Our method is computationally

<table>
<thead>
<tr>
<th>Cases</th>
<th>No. of equality constraints</th>
<th>No. of inequality constraints</th>
<th>Our method</th>
<th>KKT method</th>
<th>Speed up ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU time (seconds) (i)</td>
<td>Final objective value (100 MVA base)</td>
<td>CPU time (seconds) (ii)</td>
<td>Final objective value (100 MVA base)</td>
<td>(i) / (ii)</td>
</tr>
<tr>
<td>(1c)</td>
<td>120</td>
<td>46.72</td>
<td>846.56</td>
<td>13.3209</td>
<td>18.12</td>
</tr>
<tr>
<td>(2c)</td>
<td>150</td>
<td>48.25</td>
<td>1119.88</td>
<td>13.3310</td>
<td>23.21</td>
</tr>
<tr>
<td>(3c)</td>
<td>180</td>
<td>52.28</td>
<td>1316.93</td>
<td>13.3798</td>
<td>25.19</td>
</tr>
<tr>
<td>(4c)</td>
<td>210</td>
<td>56.31</td>
<td>1493.34</td>
<td>13.4023</td>
<td>26.52</td>
</tr>
</tbody>
</table>
efficient and has a good convergence rate.

The test results show the sequential version of our method has achieved a dramatic speed-up ratio compared with the IMSL constrained nonlinear programming subroutines in solving distributed OPF problems. Besides, our method is suitable for the implementation in a distributed computer network and can be used as a basic optimization module for handling optimization problems of large-scale distributed systems.

Acknowledgement

This research work is partially supported by National Science Council in Taiwan under grand # NSC 93-2622-E-129-004-CC3 and NSC 93-2515-S-129-004. The authors would like to thank the reviewers for their useful suggestions and comments to improve the quality of this paper.

References


Appendix

A.1 The Modified Parallel Block Scaled Gradient (MPBSG) Method [18]

The modified parallel block scaled gradient method with decentralized Armijo-type step-size rule is to solve the block additive unconstrained optimization problems. The proposed rule determines a flexible step-size for each individual subsystem to ensure the decrease of the overall objective value and executes in a large distributed computing environment.

Since $F_i$ is function of $x_i$ and $x_i^{(k)}$ in Eq. (16). The minimization problem in Eq. (16) that needs to be solved in phase 1 can be rewritten as follows:

$$
\min \sum_{i=1}^{n} \left\{ \nabla x_i F_i^T (x_i(k), x_i^{(k)}(k))dx_i + \frac{1}{2} dx_i^T H_i dx_i \\
+ \lambda_i^T [g_i(x(k)) + \nabla x_i g_i^T (x(k))dx_i] \right\} 
$$  (A-1)

which is indeed a block additive unconstrained optimization problem.

Where $x_i^{(k)}$ is the boundary state vector of the other subsystems which are connected with subsystem $i$. The $x_i$ is boundary state vector of subsystem $i$.

We define

$$
J_i(x_i, x_i^{(k)}) = \nabla x_i F_i^T (x_i(k), x_i^{(k)}(k))dx_i + \frac{1}{2} dx_i^T H_i dx_i \\
+ \lambda_i^T [\nabla x_i g_i^T (x(k))dx_i + g_i(x(k))] 
$$  (A-2)

$J(x) = \sum_{i=1}^{n} J_i(x_i, x_i^{(k)})$  (A-3)

denotes the objective function of Eq. (16).

The modified parallel block scaled gradient [18] method uses the following iteration

$$
x_i(w+1) = x_i(w) + \gamma \cdot J_i(x_i(w)) 
$$  (A-4)
to solve Eq. (A.3) for each subsystem $i$, $i = 1, 2, \ldots, n$, where $w$ is iteration index, $\gamma_i(w)$ is a positive decentralized step-size and $s_i(w)$ is a descent direction obtained from

$$M_i(x_i(w), x^i_{ib}(w)) s_i(w) = -\nabla x_i J_i(x(w)), \quad (A.5)$$

in which the predetermined matrix $M_i(x_i(w), x^i_{ib}(w))$ is chosen to be a positive definite matrix and satisfies the following condition

$$K_0 \|Z_i\|^2_{L_i} \geq Z_i^T M_i(x_i(w), x^i_{ib}(w)) Z_i \geq K_1 \|Z_i\|^2_{L_i} \quad (A.6)$$

for all $Z_i \in \mathbb{R}^N$. And some constants, $K_0 > 0$, $K_1 > 0$. And $\nabla x_i J_i(x(w))$ is the gradient of $J_i(x(w))$ with respect to $x_i$. In general, for a large distributed system, for example a power system formed by interconnected subsystem, a meaningful matrix $M_i(x_i(w), x^i_{ib}(w))$ is usually a sparse matrix which allows the linear equation Eq. (A.5) to be solved by a sparse matrix technique.

### A.2 Decentralized Armijo-Type Step-Size Rule

In centralized approach, step-size $\gamma_i(w)$ can be determined by an Armijo-rule based on the information of the whole system. However, this centralized step-size determination rule requires lots of communication overhead in a distributed computing environment. In [19], $\gamma_i(w)$ is set as a constant for all $i$, and the convergence of the parallel block scaled gradient method is guaranteed if the constant $\gamma_i(w)$ lies within certain range; However, the upper bound of the step-size range is determined on the basis of a Lipschitz constant. The Lipschitz constant is unknown and the range is very conservative. To circumvent this critical drawback in step-size determination, we propose the following two-phase decentralized rule to determine the Armijo-type step-size. Initially, we choose the following parameters, $\eta_i > 0$, $\eta_i < 1$, and set $\gamma_i(w) = 1$ at $w = 0$ for each system $i$. Then at each iteration $w$, once $s_i(w)$ is obtained from Eq. (A.5), the following two-phase decentralized rule will be performed to determine Armijo-type step-size.

Phase 1: For each subsystem $i$, we set

$$\gamma_i(w) = \gamma_i(w-1) \eta_i^{m_i(w)}, \quad (A.7)$$

where $\gamma_i(w-1)$ is the resulted step-size value after the completion of iteration $w-1$, and $m_i(w)$ is the smallest non-negative integer $m$. The $m_i$ holds the following inequality

$$J_i(x_i(w) + \gamma_i(w-1) \eta_i^{m-1} s_i(w), x^i_{ib}(w))$$

$$+ \sum_{l \in L(i)} j_i(x_i(w), x^i_{ib}(w) + \gamma_i(w-1) \eta_i^{m-1} s^i_{lb}(w))$$

$$< J_i(x_i(w), x^i_{ib}(w)) + \sum_{l \in L(i)} j_i(x_i(w), x^i_{ib}(w))$$

$$- \frac{K}{2} \gamma_i(w-1) \eta_i^{m-1} \|s_i(w)\|^2_{L_i}, \quad (A.8)$$

where $K$ is found in Eq. (A.6) and $j_i$ is the basic objective function of other subsystem bus $l$ connecting with subsystem $i$ [18].

Update of $x_i(w + 1)$: Following Phase 1, we use the obtained $\gamma_i(w)$ to update $x_i(w + 1)$ by Eq. (A.4).

Phase 2: The purpose of this phase is to adjust the step-size $\gamma_i(w)$ obtained in Phase 1. After receiving all the necessary data from neighboring subsystems, each subsystem $i$ will investigate whether the following inequality holds

$$\sum_{l \in L(i)} j_i(x_i(w+1), x^i_{lb}(w+1)) + \sum_{l \in L(i)} j_i(x_i(w), x^i_{lb}(w))$$

$$\leq \sum_{l \in L(i)} j_i(x_i(w+1), x^i_{lb}(w)) + \sum_{l \in L(i)} j_i(x_i(w), x^i_{lb}(w))$$

$$+ \frac{\sigma K_i}{2} \gamma_i(w) \|s_i(w)\|^2_{L_i} + \sum_{l \in L(i)} \frac{\sigma K_i}{2} \gamma_i(w) \|s_i(w)\|^2_{L_i}, \quad (A.9)$$

where the scalar $\sigma \in (0, \frac{2}{\max_{l \in L(i)} |L(i)|})$, and $|L(i)|$ is a known value for a given system. We then adjust the step-size $\gamma_i(w)$ obtained in Phase 1 according to the following rule:

$$\gamma_i(w) := \begin{cases} \gamma_i(w) & \text{if Eq. (A.9) holds} \\ \gamma_i(w) \eta_i & \text{otherwise}. \end{cases} \quad (A.10)$$

Moreover, if Eq. (A.9) does not hold for subsystem $i$, we also need to reduce $\gamma_i(w)$ for all $l \in L(i)$ by

$$\gamma_i(w) := \gamma_i(w) \eta_i. \quad (A.11)$$

Finally, each subsystem $i$ will send the resulting value of $\gamma_i(w)$ to all subsystem $l$’s, $l \in L(i)$, and continue the next iteration. The proposed rule determines a flexible step-size for each individual subsystem to ensure the decrease of the overall objective value and enables our method to execute in a large-scale distributed computing environment. From the above, we see that the Lipschitz constant of the objective function is not required in our decentralized step-size determination rule, the amount of data transferring between computer $i$ and computers corresponding to subsystem $l$’s, $l \in L(i)$ is very small. Only the values involved with boundary states. Consequently, a local synchronization scheme is sufficient to carry out our parallel algorithm, which can be easily implemented in a distributed computer network.
Huay Chang received the B.B.A. degree in Business Administration from Fu-Jen University, Taiwan in 1983, and the M.S. degree in Management Information System from Utah State University, USA in 1987, and the Ph.D. degree in Management from Fu-Dan University, China in 2005. She is now an Instructor in the Department of Information Management of Chihlee Institute of Technology in Taiwan. Her currently research interests are Information System, Network Technique, Wisdom Computation, Optimization Theory, Knowledge Management, Audio Recognition.

Shieh-Shing Lin received the B.E. degree in Electrical Engineering from Jung-Yuan University, Taiwan in 1983, and the M.S. degree in Electrical Engineering from Utah State University, USA in 1986, and the Ph.D. degree in Electrical and Control Engineering from Chiao-Tung University, Taiwan in 2002. He is now an Associate Professor in the Department of Electrical Engineering of St. John’s University in Taiwan. His major research interests are Optimization Theory and Large-Scale Power System.